From: Proceedings of the Eleventh International FLAIRS Conference. Copyright © 1998, AAAI (www.aaai.org). All rights reserved.

Function Modeling for an Integrated Framework: a Progress Report

Filippo A. Salustri, P.Eng.
Industrial and Manufacturing Systems Engineering
The University of Windsor
401 Sunset Ave., Windsor, Ontario, N9B 3P4, Canada
salustri@uwindsor.ca

Abstract

This paper discusses some preliminary results of the author's on-going research into the representation of product function information for integrated knowledge-based environments. Behavior is defined as what a product does, and function is defined as how the behavior is achieved. *Predicative descriptions* (of either function or behavior) are described in terms of verb-object pairs (VOPs). A VOP may represent either a function or a behavior, depending on the *context* in which the VOP appears. A discussion of the implementation of this representation in the author's DKSL knowledge-based system is also given.

Introduction

The principal thrust of the author's research is to construct an integrated knowledge-based system (KBS) that can be used universally throughout an arbitrary engineering design process. In order to provide for much-needed logical rigor, the development of the KBS is rooted in a formal logical theory of design also under development by the author. The ultimate goal is to develop an integrated, logically rigorous framework for the representation of design knowledge.

One area currently under investigation in this on-going project is the incorporation of product function modeling, especially in the initial stages of a design process. It is widely accepted that deferral of design decisions about product form, especially in the early stages of product development, is good engineering practice. But while a variety of research efforts have studied and continue to study the role of function modeling in design, relatively little effort has been invested in connecting function modeling with more conventional form-oriented design. It is this shortcoming that the current author seeks to address.

This paper will present preliminary results for this component of his overall theory of designing, including both theoretical and implementation issues. The focus of this paper will be on the synthesis of appropriate ontological concepts to support function modeling in an integrated framework. Process-related concerns (i.e. the mapping from function to form) are deferred to a future paper.

We will begin by defining certain key terms. Then, a discussion of the theory of product function modeling being developed will be given, followed by a presentation of representation and implementation matters. Finally, there will be a discussion of the current results and a statement of conclusions.

Definitions

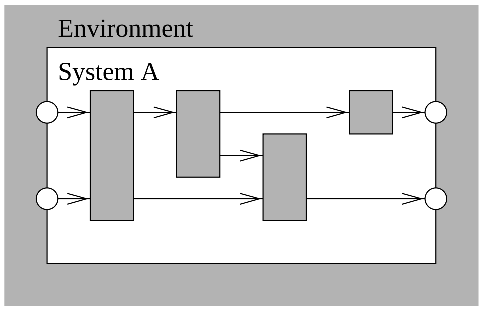
There is a substantial difference of opinion in the research community regarding the meaning of terms such as *function* and *behavior*; no formalized, standardized and consistent model of either of these terms currently exists. Some researchers consider function as a description of the actions a product can perform (e.g. (Qian & Gero 1996)), while others treat it as a description of a subset of *behaviors* (i.e. intended behavior or "purpose" – as in (Sturges, O'Shaughnessy, & Kilani 1996)). A variety of other definitions are given in (Chittaro, Tasso, & Toppano 1994; Chakrabarti 1993).

Roughly speaking, the current author uses the following definitions: *behavior* is the response of a system to stimuli, whereas *function* is how the system achieves its behavior. While this appears to invert the common definitions of these terms as found in the AI and function reasoning literature (including such areas as software engineering), it *is* consistent with the intuitions of many typical electro-mechanical designers. For example, the *behavior* of a mechanical part is generally thought of as its outward, measurable response to, say, mechanical loads; similarly, the internal phenomena that cause the behavior of an assembly is commonly described as how the assembly *functions*. Since the author's work is being carried out in this kind of environment, and is intended primarily for the electro-mechanical engineering community, this distinction will be assumed for the rest of this paper.

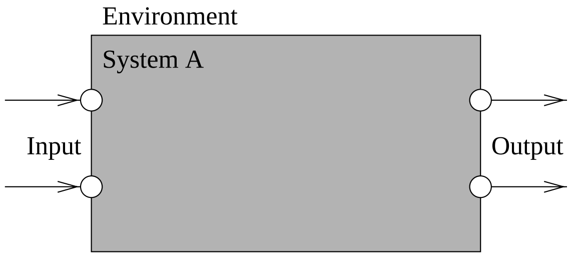
Furthermore, the author also distinguishes both function and behavior from the terms *performance* and *operation*; each term denotes a particular kind of design knowledge, and they are all important. Let us now consider more closely the interrelationship between these terms.

Behavior is defined as the response of a system to a given set of inputs (which are not necessarily specified quantitatively); it describes the *role* played by the system in a larger entity. The behavioral perspective takes the product being designed to be a "black box" whose internal function is neither visible nor known; the inputs, outputs, and operating environment of the product, on the other hand, are "transparent" and known. This is shown graphically in Figure 1(b).

Insofar as behavior describes the response of a product to certain inputs, it is seen as answering the question *"What does the product do?"* Behavior is described without commitment to the form of the product. By *operational environment*, we mean the environment in which the product, once fabricated, is expected to operate.



(a) Example Functional Perspective



(b) Example Behavioral Perspective

Figure 1: Perspectives of function and behavior

Function, on the other hand, is seen as a description of *how* a product works rather than what it does. This is shown graphically in Figure 1(a), where the environment is now opaque, and the product is "transparent," and is composed of a series of black box subsystems whose interaction describes how the product comes to exhibit a certain behavior, but without necessarily making commitments about the form of the product.

Function and behavior are distinguished from performance and operation. *Performance* is quantifiable measure of how well a product meets its operational, functional, behavioral, and other design requirements. *operation* is defined as the time-dependent sequencing of events occurring during the product's use which trigger, or are triggered by, the product's functions and behaviors. Since the author's current focus is on function and behavior, matters of performance and operation will be set aside for the rest of this article.

Function and Behavior

Sometimes, confusion can arise when considering these terms – function and behavior – because they may both be said to be "functional," rather than structural, descriptions of products (e.g. functional functions versus functional behaviors). Also, it is often possible to represent both functions and behaviors in single natural language clauses that seem quite intuitive to humans: consider the phrase "...to support a load in bending...." Both behavior ("to support a load") and function ("in bending") are intimately connected in a single phrase. The fact that both the behavioral and functional aspects of this case can be described in a single natural language statement only obscures their distinction. This constitutes, in the author's opinion, a significant problem with the use of natural language – indeed, any *in*formal language – to precisely define the nature of designed product function.

The author prefers the term "predicative," rather than "functional," to capture the commonality of the terms *function* and *behavior*. Firstly, while distinguishing them from other terms having to do with the structure of products, it is not as ambiguous as the term "functional." Secondly, the term seems appropriate because the expression of both function and behavior descriptions in natural languages tend to be formed as *predicate* clauses consisting of verb/object pairs (VOPs). That is, both the above examples describe an action performed by an entity upon some other entity, independent of the phrasing in natural language. The importance of VOPs will be further discussed below.

It is important to note that the author does not intend to imply a formalization based strictly on linguistic considerations. Instead, the notion of a VOP is used only to admit that predicative descriptions are complex structures have an active component (the "verb" part) and a descriptive/structural component (the "object").

In order to continue the examination of predicative descriptions, let us consider the following four statements:

1. The refrigerator keeps food cold.
2. The refrigerator keeps things cold.
3. The refrigerator preserves food.
4. The refrigerator lowers ambient temperature in an enclosed space.

Any of these statements in isolation can be considered a *behavior* of a refrigerator; i.e. a response of the product to certain inputs. If we consider statement #1 as a behavioral description, we may then legitimately ask *"How is this behavior achieved?"* One possible answer is given by statement #4, which involves isolating a region of space, transferring heat from that space by some means, etc. So, statement #4 is a functional description related to the behavioral description in statement #1.

We may also ask *"Why does the refrigerator keep food cold?"* One answer to this question is statement #3. In this case, statement #1 describes a function rather than a behavior, and statement #3 describes a behavior with respect to statement #1.

In other words, any of the four statements can be taken as descriptions of either function or behavior, depending on the perspective taken during their utterance. Predicatives are therefore *relative to the reference frame of an agent*, in which statements about a product are being made; they are thus *not* intrinsic properties of designed products.

Furthermore, the reference frame in which a predicative statement is made, contains definitions that allow semantic interpretation of statements; that is, they are *contexts*. Clearly, then, context plays a central role in function modeling, in that it (a) provides terminological information about the definitions of words appearing in the statements, and (b) implies a great deal of information about the operational environment. For example, in statement #1, terms "cold" and "keep" are relative to a context of food refrigerators; these definitions *may* be the same in other contexts, but this is not necessarily the case in every other context. There are various on-going efforts to formalize the notion of context; an excellent overview is available in (Akman & Surav 1996).

The author's views on function and behavior can be summarized in the Table 2.

Behavior	**Function**
role-dependent	operational
goal-oriented	process-oriented
what a system does	how a system does it
based on purpose/usage	based on physical properties

Figure 2: Summary of function and behavior

The apparent verb/object structure of both functions and behaviors has been used as the root of various formalizations (e.g. (Umeda *et al.* 1996)); the current author also employs this approach. Consider again some statements about a refrigerator, this time depicted graphically in Figure 3. Statements 1 and 3 are related through function/behavior relations. Statement 2 is related to statement 1 by generalization on the object of the VOP (i.e. "food" and "things"). Note that a similar generalization carried out on statement 2 (yielding statement 4) is relatively meaningless. The author contends that generalizations will not transfer through how/why relations; in other words, combinations of abstraction relations and predicative relations are not transitive. Furthermore, that the abstraction occurred only on the object part of the VOP suggests that abstraction can occur on either of the verb or object parts of a VOP independently. (As an example of generalization on verbs, consider the verb form "to move," and its specialization "to shift.") Finally, while VOP abstraction can occur on either the verb or the object parts, only the verb part is needed for the predicative relations. Thus, a representation of predicative relations may give priority to the verb part of VOPs without loss of generality or expressiveness. We will take advantage of this observation in the next section.

Because of its relative popularity, one other particular viewpoint regarding function and behavior bears discussion here. A number of researchers suggest that functions should be derived from one of three primitive *transfers*: those of mass, of energy, and of information (originated by Rodenacker in (Rodenacker 1971)). There are two problems with this approach. First, the notion of *storage* is not represented; though one might consider a mass transfer of zero magni-

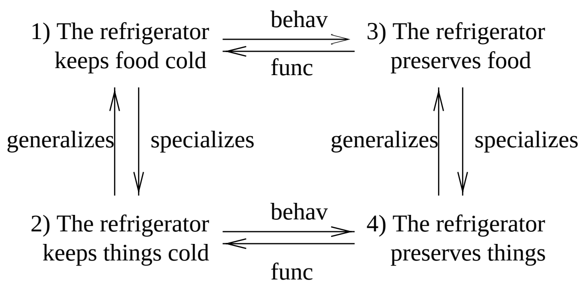

1) The refrigerator keeps food cold
behav
func
3) The refrigerator preserves food
generalizes
specializes
generalizes
specializes
2) The refrigerator keeps things cold
behav
func
4) The refrigerator preserves things

Figure 3: Abstraction of function and behavior

tude as a representation of mass storage, it is also equivalent to – and thus indistinguishable from – a complete lack of transfer. Second, this approach does not seem to treat *transformations*. For example, when a beam is subjected to mechanical loading, incoming energy is transferred to the beam, where it is *transformed* into strain energy, and then essentially *stored* there until the environmental conditions are such that the energy can be released (e.g. by removal of the applied load).

Although it is possible to represent the transformation of energy in the above example as a series of transfers, eventually reaching down to the atomic level, this is not always so, nor is it necessary in typical design situations. For example, the emission of thermal energy by an object that has been exposed to microwave energy requires a transformation in the frequency of the energy; it is unclear how such a transformation can be represented by transfers alone.

The three basic transfers are considered disjoint, but also have a uniform representation – typically a vector whose magnitude and direction represent the amount and direction of transfer, respectively. Extending this representation for storages is straight-forward by using a scalar quantity (as opposed to zero-magnitude vectors, which stand for zero-rate transfers). Transformations, on the other hand, are not so easily treated. This is a matter currently being studied by the author, but the notion of using tensors may hold some promise here. Alternatively, it may be possible to use a representation similar to that of bond graphs (Karnopp, Margolis, & Rosenberg 1990) to achieve the same end.

In any event, this approach tends to attach concise, well-defined, and (generally) broadly applicable semantics to the primitive transfers. The alternative is to allow *ad-hoc* predicative descriptions; in this case, the inherent loss of semantic rigor is made up for by a potentially providing descriptions better suited to particular situations, without requiring very "deep" hierarchies. It is unclear at this time which approach is the best overall.

Representing predicative descriptions

Since our goal is to represent predicative descriptions so as to allow their inclusion in and manipulation by knowledge-based systems, we must establish some means of capturing that information in a structured and logical fashion.

The representation under development extends a frame-based knowledge representation system of the author's design, called DKSL, written in the Scheme programming language, and which is intended to support product modeling for designed artifacts. Frames provide a richer, more expressive way of capturing design knowledge than do typical object-based systems. The most fundamental difference between objects and frames is that while data is assigned to object/attribute pairs in object-based systems, data in frame systems is assigned to frame/slot/facet triplets. This extra naming level in frame-systems significantly extends the kinds and amounts of data that can be associated with entities. Another fundamental difference between frames and objects is that object classes can model function (in this author's sense) of the underlying objects, while frames use procedural attachments – so-called *demons* – to capture dynamic aspects of computation. Nonetheless, it *is* possible to create a class-based object system using frames, as has been done by Mark Kantrowitz in his FrameWork system. Thus, this author feels that frame-based systems are actually more flexible and richer than typical object-oriented systems.

Additionally, DKSL is specifically tailored to the needs of engineered product modeling. Among DKSL's distinguishing features are: support for contexts, allowing multiple definitions of terms; use of prototypes (or *exemplars*) rather than classes; and uniform semantics of slots (i.e. relations) over whole contexts. DKSL's ontology currently supports notions of quantity, features, parts, assemblies, subassemblies (or subsystems), as well as types of all these.

In order to include predicative descriptions in DKSL, the author is extending the DKSL ontology structure. This section gives an overview of those extensions.

Part of the structure of the resulting frame system for the refrigerator example is shown graphically in Figure 4; frames and slots are shown in capitals, and facets are shown as labeled links between them. Unlabeled links connect frames to slots.

Verb parts of VOPs are implemented as relations (i.e. slots) between frames; thus, predicatives become inherent parts of the descriptions of product models in DKSL. Each such relation is a specialization of a generalized "verb" entity. Searching for explicitly defined functions or behaviors is a simple matter of extracting those slots in a frame that are specializations of the appropriate verb exemplar.

Verb slots anchor predicative descriptions. They have various facets which hold information that further specify a given instance of a verb slot. In the refrigerator example (refer to Figure 4, a *how* facet associates the verb slot "keeps" with the adverbial descriptor "cold," and a *what* facet associates "keeps" with the entity "food." In its final form, DKSL will support other kinds of facets (e.g. "when" and "why" facets); these are currently being studied and developed.

The *func* relation is implemented as a facet on verb slots, and connects a behavior to a function; the *behav* relation is its inverse. However, since a predicative description can be either a function or a behavior, verb slots are not distinguished as functional or behavioral; rather, a predicative description is viewed as functional or behavioral depending on which of *behav* or *func* facets is being considered. A behavior can be implemented by a number of functions work-

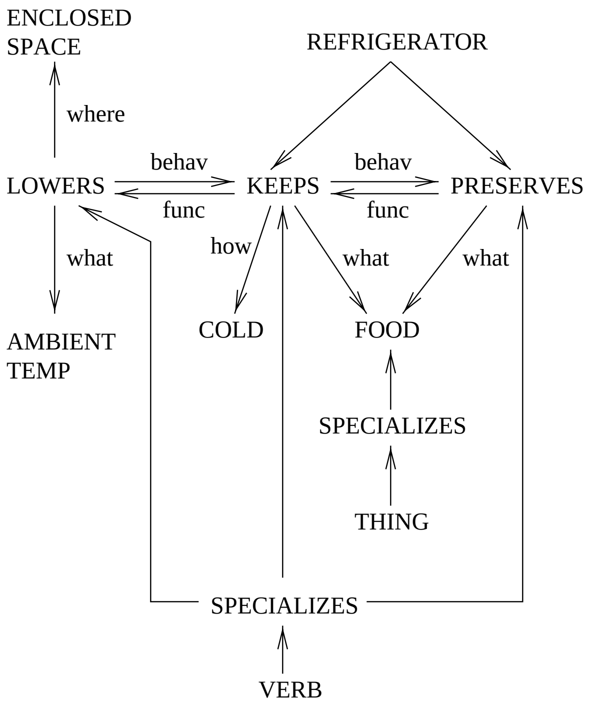


Figure 4: Sample knowledge structure in DKSL

ing together; therefore, a *func* link is a one-to-many relation, where the behavior arises from the conjunction of the functions indicated by a *func* link. Disjunction of functions is considered meaningless. A similar argument is made for *behav* links.

This representation of function and behavior will be used to study and develop reasoning algorithms that can use predicative descriptions. One obvious example is the potential application to case-based reasoning (Riesbeck & Schank 1989). Assuming a knowledge-base of cases containing descriptions of function and behavior as well as product structure, a case-based reasoning system can use *functional requirements*, commonly defined in the very early stages of product design processes, to suggest alternative configurations and designs that may be suitable for a given task. Functional knowledge may also be used to develop failure mode error analyses of products, as well as to develop and study the operational procedures for product use. If the author's overall theory can be extended into the area of design process management, predicative descriptions may find a role in the analysis and improvement of organizational performance of design enterprises.

Conclusions

The foregoing sections have presented the author's preliminary results, summarized below, in the development of a KBS component for modeling product function and behavior. The focus of this paper has been on the theoretical description of functions and behavior rather than on implementation issues. The author is currently beginning the implementation phase of the model as presented above, as a module of the DKSL system for design knowledge (other portions of which have been discussed elsewhere (Salustri 1996a; 1996b) in the form of the *Designer* system).

Design concepts are viewed as predicative – rather than structural – descriptions, and specialization is based on constraints on function and behavior, rather than on form (per (Qian & Gero 1996)). Predicative descriptions are rooted on the verb slots used to describe the function or behavior, with further specifications defined via facets on those slots. This perspective is a fundamental aspect of the KB implementation in order to promote thinking about function rather than structure in the early design stages. The author's eventual goal is to provide a computer-based system with a graphical user interface that will allow interconnectivity between predicative-based design and other, form-based design methodologies. The central role of contexts in function modeling has been identified; they provide the apparatus by which the author will be able to formalize the reasoning processes that use information about function.

Clearly, a great deal of work remains to be done, but the author is confident that some interesting results from the implementation phase will be available within the next year.

Acknowledgments

The author gratefully acknowledges the National Sciences and Engineering Research Council of Canada for funding this work under grant number OGP0194236.

References

Akman, V., and Surav, M. 1996. Steps toward formalizing context. *AI Magazine* 17(3):55–72.

Chakrabarti, A. 1993. Towards a theory for functional reasoning in design. In Roozenburg, N. F. M., ed., *Proceedings of ICED 93, 9th International Conference on Engineering Design*, volume 1, 1–8. Zurich, Switzerland: Heurista.

Chittaro, L.; Tasso, C.; and Toppano, E. 1994. Putting functional knowledge on firmer ground. *Applied Artificial Intelligence* 8:239–258.

Karnopp, D. C.; Margolis, D. L.; and Rosenberg, R. C. 1990. *System Dynamics: A Unified Approach*. New York: Wiley and Sons.

Qian, L., and Gero, S. 1996. Function-behavior-structure paths and their role in analogy-based design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 10:289–312.

Riesbeck, C. K., and Schank, R. C. 1989. *Inside Case-Based Reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Rodenacker, W. 1971. *Methodisches Konstruieren*. Berlin: Springer-Verlag.

Salustri, F. A. 1996a. A formal theory for knowledge-based product model representation. In *Knowledge-Intensive CAD II: proceedings of the IFIP WG 5.2 workshop*. Chapman & Hall.

Salustri, F. A. 1996b. Integrated computer modeling of engineering design information. In Meguid, S. A., ed., *Proceedings of the 1996 Forum of the Canadian Society for Mechanical Engineering*, 613–624.

Sturges, R. H.; O'Shaughnessy, K.; and Kilani, M. I. 1996. Computational model for conceptual design based on extended function logic. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 10:255–274.

Umeda, Y.; Ishii, M.; Yoshioka, M.; Shimomura, Y.; and Tomiyama, T. 1996. Supporting conceptual design based on the function-behavior-state modeler. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 10:275–288.